The ISC Int'l Journal of Information Security

July 2024, Volume 16, Number 2 (pp. 1–11)
http://www.isecure-journal.org

Customizable Utility-Privacy Trade-Off: A Flexible Autoencoder-Based Obfuscator

Mohammad A. Jamshidi[1,*], Mohammad M. Mojahedian[1], and Mohammad R. Aref[1]
[1]*Information Systems and Security Lab. (ISSL), Sharif University of Tech., Tehran, Iran*

A R T I C L E I N F O.

Article history:
Received:
Revised:
Accepted:
Published Online:

Keywords:
Autoencoder, Collaborative Learning, Deep Neural Networks, Privacy-Utility Trade-Off

Type: Research Article

doi:

dor:

Abstract

To enhance the accuracy of learning models, it becomes imperative to train them on more extensive datasets. Unfortunately, access to such data is often restricted because data providers are hesitant to share their data due to privacy concerns. Hence, it is critical to develop obfuscation techniques that empower data providers to transform their datasets into new ones that ensure the desired level of privacy. In this paper, we present an approach where data providers utilize a neural network based on the autoencoder architecture to safeguard the sensitive components of their data while preserving the utility of the remaining parts. More specifically, within the autoencoder framework and after the encoding process, a classifier is used to extract the private feature from the dataset. This feature is then decorrelated from the other remaining features and subsequently subjected to noise. The proposed method is flexible, allowing data providers to adjust their desired level of privacy by changing the noise level. Additionally, our approach demonstrates superior performance in achieving the desired trade-off between utility and privacy compared to similar methods, all while maintaining a simpler structure.

1 Introduction

Recent progress in deep neural networks has been fueled by having a lot of data and powerful computers. However, sometimes, access to specific data is restricted, which can slow down progress. Specifically, some data providers are cautious about sharing sensitive information due to privacy concerns. For example, think about a group of medical researchers who want to create a predictive model for identifying early signs of a particular disease using patient health records from different hospitals. However, the hospitals are careful about patient privacy and do not want to share the actual patient data. This leads to the need to find ways to protect patient privacy while still using their data to create a useful model. As another example, financial institutions with extensive transaction records are often constrained by regulations that limit their ability to share customer information. This balance between harnessing the potential of big data and safeguarding privacy rights presents a complex and ongoing challenge in the era of machine learning, necessitating the exploration of ways to protect users' privacy while still utilizing their data to create valuable models.

Privacy concerns can be tackled through two primary types of learning methods: federated learning and centralized learning. Federated learning, gaining notable attention, is a decentralized approach that al-

* Corresponding author.
Email addresses: `m.a.jamshidi992@gmail.com`, `m.mojahedian@gmail.com`, `aref@sharif.edu`
ISSN: 2008-2045 © 2024 ISC. All rights reserved.

lows multiple devices to collaboratively train a shared model. It addresses privacy concerns by keeping sensitive data localized; however, challenges related to communication efficiency and model convergence may arise [1–3]. It is noteworthy that while federated learning ensures that raw data is not directly shared with users, the model may still retain information from the dataset. Additionally, federated learning remains susceptible to both membership inference and poisoning attacks [4–7]. On the contrary, centralized learning follows a more conventional approach where data from various sources is gathered and stored in a central server for model training. Despite concerns regarding privacy and data security, centralized learning offers several benefits, including improved computational efficiency, enhanced model accuracy, and simplified model deployment. This paper will specifically delve into investigating the privacy issue within the context of centralized learning.

Let us consider a scenario where a utility provider requests datasets from various data providers. In centralized learning, the key concept for achieving privacy is for data providers to obfuscate their data in a manner that ensures the desired level of privacy from both the utility provider and potential adversaries. Simultaneously, the utility provider should have the capability to efficiently use the distorted data for training the learning model. The resulting model can then be employed by the utility provider to offer services to data providers or other users. Figure 1 illustrates the schematic of the obfuscation process. The primary objective of different obfuscation techniques is to strike a balance between data utility and privacy considerations. Numerous algorithms have been proposed in the literature to obfuscate datasets. The primary techniques introduced to safeguard privacy encompass k-anonymity [8], ℓ-diversity [9], and t-closeness [10], tailored for smaller datasets. In addition, approaches based on Homomorphic Encryption (HE) [11, 12] and Secure Multi-party Computation (SMPC) [13, 14] are hindered by substantial computational and communication overheads, making them less common in practical use. Another cluster of privacy-preserving algorithms hinges on differential privacy, a mathematical tool that guarantees data privacy by introducing suitable noise [15–17]. However, differential privacy encounters challenges in higher dimensions due to its potential time consumption and the need for significant noise addition, which may subsequently distort the utility of the dataset.

Neural network-based obfuscators have emerged as an alternative approach that has gained significant attention, especially with the increasing availability of data in recent years [18–26]. These approaches leverage various machine learning techniques, including autoencoders, generative adversarial networks (GANs), and variational autoencoders, to accomplish their objectives. Below, we provide more details on several noteworthy works within this field.

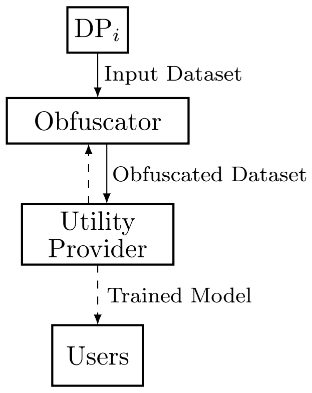

Figure 1. Data providers (DPs) aim to find a balance between privacy and utility by altering their data before sharing it with the utility provider.

Huang *et al.* [18, 19] employ GANs and formulate a minimax game between the privatizer and the adversary to conceal a specific feature. Li *et al.* [20] utilize mutual information-based training to learn a feature extractor that hides private features while preserving the utility of the remaining information. However, the output of their method is in a censored feature vector format, which is incompatible with the original data format and unsuitable for data publishing or pre-trained models like DenseNet. In [21], Singh *et al.* combine variational autoencoders and differential privacy techniques. They first reduce the data dimension, separate sensitive and non-sensitive data using two classifiers, and then perturb the covariance matrix of the sensitive data to establish differential privacy. However, utilizing differential privacy methods typically requires more training data and is time-consuming [27]. Osia *et al.* propose a feature extraction method in [23] specifically designed for implementation on mobile devices using a Siamese architecture. However, this method also generates data that does not adhere to the original dataset format. In [24], Nguyen *et al.* employ an autoencoder to reduce image dimensions and then utilize a GAN-based structure to make the distribution of encoder outputs approach a Gaussian distribution. Additionally, the classifier providing the desired feature offers feedback to the obfuscator. Mandal *et al.* [25] propose a private learning algorithm based on the uncertainty autoencoder.

In this paper, we propose that data providers possess a specific feature in their datasets that necessitates protection against unauthorized access. To achieve this, each data provider employs an obfuscation procedure, depicted in Figure 1, to generate a dataset that ensures privacy for the specific feature while preserving utility for other features. The obfuscator, which is a neural network, can be trained by either the utility provider or another certified entity and then distributed to the data providers for implementation. The fundamental concept underlying the obfuscation framework is illustrated in Figure 2. Initially, we employ data compression with two primary objectives:

(1) Reducing the complexity of the intermediate networks, thereby decreasing the number of parameters and enabling effective operation with smaller data volumes.
(2) Generating data instances with reduced correlation.

By incorporating a classifier into the intermediate network framework, we extract the desired private feature for obfuscation. By reducing correlation, the compression process guarantees that obfuscating the output of the classifier has minimal influence on other features. Furthermore, we employ an additional neural network to ensure there is no correlation between the private feature and other information. The compression structure utilized is an autoencoder. Once we have ensured that the private feature is effectively uncorrelated with other information in the intermediate layers, we proceed to add suitable noise to the private feature to conceal it. The proposed scheme demonstrates superior performance compared to similar schemes while maintaining a simpler structure without relying on GANs or other feedback-based structures, which guarantees stability [28–30].

The remainder of the paper is organized as follows: Section 2 describes the system model, Section 3 provides details of the proposed methodology, Section 4 presents simulation results, Section 5 explores some additional privacy considerations, and finally, Section 6 concludes the paper.

2 System Model

The dataset $\mathcal{D}$ comprises n samples from the instance space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ represents the input space and $\mathcal{Y}$ denotes the output space. In this context, the vector $\mathbf{Y} \in \mathcal{Y}$ encompasses various features, which are categorized as private and non-private, denoted as $\mathbf{Y} = (\mathbf{Y}_{\mathsf{P}}, \mathbf{Y}_{\mathsf{NP}})$. To illustrate, this paper considers a collection of face images as $\mathcal{X}$, with the gender feature classified as private, while other features are considered non-private.

The data provider's objective is to share the data with the utility provider for collaborative learning. However, the data provider also places importance on maintaining the confidentiality of certain features. To accomplish this, the data provider converts $\mathbf{X}$ into $\mathbf{X}'$, to preserve the privacy of one (or several) specific features. The transformation function from $\mathbf{X}$ to $\mathbf{X}'$ is referred to as an obfuscator, and the level of ambiguity it introduces to the dataset determines the trade-off between utility and privacy.

To develop our obfuscator, we adopt a two-step approach. Firstly, we decorrelate private and non-private features. Then, we introduce noise exclusively to the private features while preserving the non-private information intact. Figure 3 illustrates our implementation using an autoencoder, which serves to reduce the data's dimensionality. This compression process yields several advantages:

(1) Enhanced Privacy: The dataset's privacy is enhanced through the dimensionality reduction achieved by the autoencoder, although this comes at the cost of reducing its utility.
(2) Decreased Correlation: The encoder's compression eliminates redundancy within the data, leading to a nearly uncorrelated output. Consequently, modifications made to the private features have minimal impact on the non-private information.
(3) Streamlined Complexity: By reducing the dimensionality through the autoencoder, the structure of intermediate networks is simplified, leading to improved efficiency.

Our obfuscation technique employs an autoencoder, consisting of an encoder E_α and a decoder D_β. The coded data, denoted as $\mathbf{V}$, is obtained by applying the encoder to the input data $\mathbf{X}$, i.e., $\mathbf{V} = \mathsf{E}_\alpha(\mathbf{X})$. Both the encoder and decoder are Deep Neural Networks (DNNs) with parameter sets α and β, respectively. Next, we generate two sets of data based on the reduced-dimensional and meaningless coded data $\mathbf{V}$:

(1) C_θ with parameter θ: This DNN acts as a classifier, taking the meaningless data $\mathbf{V}$ as input and producing meaningful features $\mathbf{W}_{\mathsf{P}} = \mathsf{C}_\theta(\mathbf{V})$. These features represent the private information we aim to protect.
(2) R_ϕ with parameter ϕ: The output of this DNN contains information from $\mathbf{V}$ that is appropriately uncorrelated with $\mathbf{W}_{\mathsf{P}}$. We denote this portion of information as $\mathbf{W}_{\mathsf{NP}}$, which we want to retain as useful as possible.

In the subsequent step, we introduce noise to $\mathbf{W}_{\mathsf{P}}$ to obtain the modified data $\mathbf{W}'_{\mathsf{P}}$. Combining $\mathbf{W}'_{\mathsf{P}}$ and $\mathbf{W}_{\mathsf{NP}}$ yields $\mathbf{V}'$. Finally, the decoder converts $\mathbf{V}'$ back to $\mathbf{X}'$. To accomplish this, we need to design appropriate parameters α, β, ϕ, θ, and choose a suitable method for introducing noise. To balance the trade-off between privacy and utility, we consider the following properties:

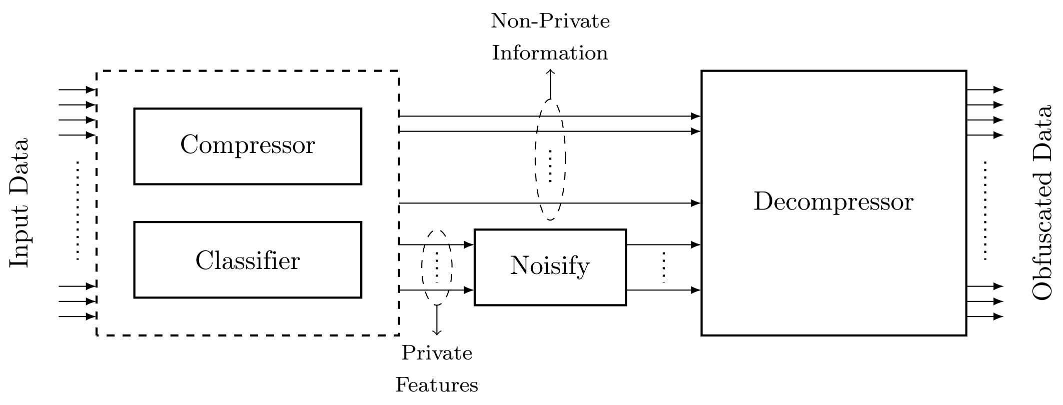

Figure 2. Through the combination of a compressor and classifier, we ensure the separation of the private feature from other information while inducing uncorrelation between them. As a result, the private feature becomes noisy while having a minimal impact on other features.

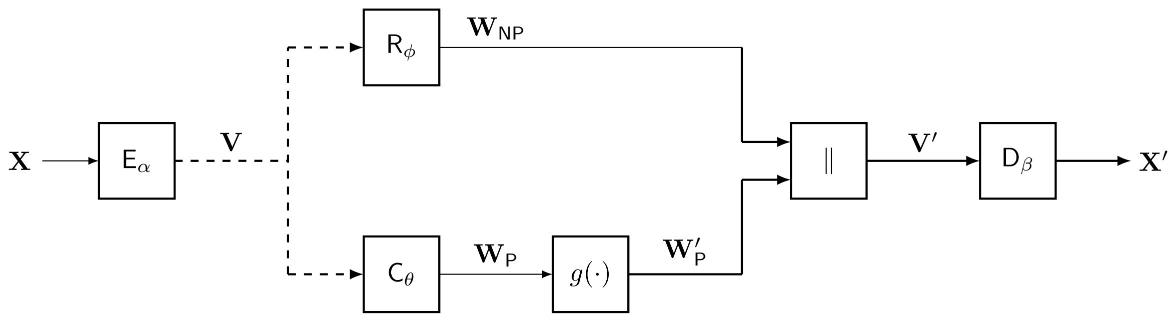

Figure 3. The obfuscator comprises an autoencoder that intelligently separates the private and non-private features in conjunction with a classifier. The private features are then appropriately combined with noise, while the non-private features remain unchanged. Subsequently, the decoder reconstructs the obfuscated dataset using all the manipulated features.

(1) **Maintain Utility:** We aim to preserve the utility of the original dataset by minimizing a loss function

$$L_{ae}(\alpha, \beta, \phi, \theta) = \mathbb{E}\Big[\ell(\mathbf{X}, \mathbf{X}')\Big]. \quad (1)$$

Here, the expectation is taken over the distribution of $\mathbf{X}$. Moreover, excluding the noise injection procedure, $\mathbf{X}'$ can be expressed as

$$\mathbf{X}' = \mathrm{D}_\beta\Big(\mathrm{C}_\theta(\mathbf{V}) \parallel \mathrm{R}_\phi(\mathbf{V})\Big). \quad (2)$$

(2) **Private feature extraction:** We seek to effectively separate the private feature from the remaining information. To achieve this, the classifier C_θ should accurately extract the private features, indicated by $\mathbf{W}_{\mathsf{P}}$. This property can be quantified by a loss function

$$L_{\mathsf{C}}(\alpha, \theta) = \mathbb{E}\Big[\ell\left(\mathbf{W}_{\mathsf{P}}, \mathbf{Y}_{\mathsf{P}}\right)\Big], \quad (3)$$

where the expectation is taken over the joint distribution of $\mathbf{X}$ and $\mathbf{Y}$.

(3) **Ensuring Decorrelation:** To ensure minimal correlation between the private feature $\mathbf{W}_{\mathsf{P}}$ and non-private information $\mathbf{W}_{\mathsf{NP}}$, we incorporate a decorrelation loss function denoted as

$$L_{\mathsf{dcor}}(\alpha, \phi) = \mathbb{E}\Big[\ell\left(\mathbf{W}_{\mathsf{P}}, \mathbf{W}_{\mathsf{NP}}\right)\Big], \quad (4)$$

where the expectation is taken over the distribution of $\mathbf{X}$.

The proposed network design consists of three stages. In the initial step, assuming no noise introduction and utilizing the mentioned properties, we determine the network parameters as

$$\begin{aligned} &\alpha^*, \beta^*, \phi^*, \theta^* \\ &\quad = \arg \min_{\alpha,\beta,\phi,\theta} L_{ae}(\alpha, \beta, \phi, \theta) + L_{\mathsf{C}}(\alpha, \theta). \end{aligned} \quad (5)$$

Next, by freezing $\mathsf{E}\alpha$ and C_θ, we ensure decorrelation while updating the parameters β and ϕ as

$$\beta^{**}, \phi^{**} = \arg \min_{\beta,\phi} L_{ae}(\alpha, \beta, \phi, \theta) + L_{\mathsf{dcor}}(\alpha, \phi). \quad (6)$$

Subsequently, to maintain the confidentiality of the private features in the final dataset, we introduce some noise to them.

To assess the effectiveness of our proposed scheme, we subject the output obfuscated dataset to evaluation by both the adversary and the utility provider, each with specific characteristics.

Adversary: We consider the adversary as a DNN whose objective is to extract a private feature from the obfuscated dataset. Specifically, we investigate an adversary described in [31], where the adversary possesses an obfuscation model and utilizes it to create an obfuscated dataset by modifying a dataset that resembles the original one. The adversary has access to both a dummy obfuscated dataset and the precise value of the feature it intends to infer. Following that, it trains its network using supervised learning techniques.

Utility Provider: The utility provider is a DNN classifier trained on the obfuscated dataset, enabling it to infer one or more features. The purpose of this network is to be made available to other users, allowing them to make inferences from their unobfuscated data.

3 Technical Details

We selected the CelebA dataset as our dataset of choice [32]. CelebA comprises a comprehensive collection of facial images, consisting of 202,599 face images from 10,177 individuals. Each image is associated with 40 binary feature labels, including attributes such as gender, age, and smile. For our specific purpose, we utilized 162,752 samples from the CelebA dataset for training, while the remaining samples were designated for testing and validation.

In our study, we considered the gender feature as private and focused on preserving the dataset's utility with respect to other features. Hence, we evaluated the dataset's utility specifically in relation to the smiling feature.

Our DNN structures were inspired by the VGG-16 network [33]. The autoencoder components, denoted as E_α and D_β, consist of four 2D-convolutional layers, three batch normalization layers, and one fully connected layer. E_α takes images of size $64 \times 64 \times 3$ as input and produces 1024 features as output, resulting in a feature vector $\mathbf{V}$ of size 1024. The decoder, D_β, converts a manipulated feature vector of size 1024 back into a $64 \times 64 \times 3$ image.

The autoencoder incorporates two fully-connected networks, C_θ and R_ϕ, in its middle layers. C_θ is a four-layer fully-connected network that maps the input of size 1024 to two outputs representing the private feature. On the other hand, R_ϕ is a three-layer fully-connected network that maps the input of size 1024 to 1022 outputs, representing the remaining uncorrelated features with the private feature. The DNNs' structures are described in more detail in Table 1.

Regarding the loss functions utilized in our approach, we employ the mean square error (MSE) for L_{ae} (autoencoder loss), the negative log-likelihood for L_{C} (classifier loss), and cross-covariance to ensure the uncorrelation between the outputs of R_ϕ and the outputs of the classifier. More specifically, we have

$$L_{ae} = \frac{1}{N}\sum_{j=1}^{N}\left(X_j - X'_j\right)^2, \tag{7}$$

$$L_{\mathsf{C}} = -\sum_{j=1}^{M} Y_{\mathsf{P},j}\log\left(W_{\mathsf{P},j}\right) + \left(1 - Y_{\mathsf{P},j}\right)\log\left(1 - W_{\mathsf{P},j}\right), \tag{8}$$

$$L_{\mathsf{dcor}} = \left\| \frac{1}{B}\sum_{j=1}^{B}\left(\mathbf{W}_{\mathsf{P}}^{(j)} - \boldsymbol{\mu}_{\mathsf{P}}\right)\cdot\left(\mathbf{W}_{\mathsf{NP}}^{(j)} - \boldsymbol{\mu}_{\mathsf{NP}}\right)^{\mathsf{T}} \right\|_{\mathsf{F}}. \tag{9}$$

Here, the dimensions of the vectors $\mathbf{X}$ and $\mathbf{Y}_{\mathsf{P}}$ are represented by N and M, respectively. The operators $(\cdot)^{\mathsf{T}}$ and $\|\cdot\|_{\mathsf{F}}$ refer to the transpose and Frobenius norm, respectively. The cross-covariance is computed over a batch of size B using

$$\boldsymbol{\mu}_{\mathsf{P}} = \frac{1}{B}\sum_{j=1}^{B}\mathbf{W}_{\mathsf{P}}^{(j)}, \tag{10}$$

$$\boldsymbol{\mu}_{\mathsf{NP}} = \frac{1}{B}\sum_{j=1}^{B}\mathbf{W}_{\mathsf{NP}}^{(j)}. \tag{11}$$

In the paper, M is set to 2, but extending it to accommodate more private features is straightforward by considering a classifier with additional outputs. It is important to mention that both the utility provider and the adversary are trained using the negative log-likelihood loss function.

The training phase of the framework is outlined in Algorithm 1. In the first step, we update the parameters of the DNNs for n_e epochs using randomly selected mini-batches of size B and the loss function described in (5). We set $B = 64$ and stopped the first step after $n_e = 50$ epochs, based on the trend of validation loss. In the second step, for n'_e epochs, we update the parameters of R_ϕ and D_β using mini-batches of size $B = 64$ and the loss function in (6), while E_α and C_θ remain fixed. Since the autoencoder and decorrelation losses are unbalanced, we consider a weighted combination of them to strike a balance that allows each loss function to contribute effectively to the overall training. Based on the values of each loss in a test iteration, we chose loss weights $w_{ae} = 0.99999$ and $w_{\mathsf{dcor}} = 10^{-5}$. We terminated the second step after $n_w = 37$ iterations, where the decorrelation loss no longer decreased significantly. The subsequent steps followed a similar procedure as the second step but with different loss weights. In each step, training was stopped when the decorrelation loss approached its minimum value while keeping the autoencoder loss below an acceptable threshold. We conducted training in multiple steps, and the best results were achieved after the third step. In the third step, we selected loss weights $w'_{ae} = 0.9999$ and $w'_{\mathsf{dcor}} = 10^{-4}$, and training was stopped after 23 iterations. It is im-

Table 1. DNNs architecture details for image datasets

Component	Num	Layer	Output Size	Specs	Activation Function
Input Data		Image Samples	$3 \times 64 \times 64$		
Encoder	1	Conv2D	$64 \times 32 \times 32$	kernel=4, stride=2, padding=1	LeakyReLU
	2	Conv2D	$64 \times 16 \times 16$	kernel=4, stride=2, padding=1	
	3	BatchNorm2D		eps=1e-5, momentum=0.1	LeakyReLU
	4	Conv2D	$64 \times 8 \times 8$	kernel=4, stride=2, padding=1	
	5	BatchNorm2D		eps=1e-5, momentum=0.1	LeakyReLU
	6	Conv2D	$128 \times 4 \times 4$	kernel=4, stride=2, padding=1	
	7	BatchNorm2D	$128 \times 4 \times 4 \rightarrow 2048$ (shaped)	eps=1e-5, momentum=0.1	LeakyReLU
	8	Linear	1024		LeakyReLU
Classifier	1	Linear	1024	Dropout (p=0.5)	LeakyReLU
	2	Linear	256	Dropout (p=0.5)	LeakyReLU
	3	Linear	64		LeakyReLU
	4	Linear	2		LogSoftMax
R	1	Linear	1024	Dropout (p=0.5)	LeakyReLU
	2	Linear	1024	Dropout (p=0.5)	LeakyReLU
	3	Linear	1022		LeakyReLU
Decoder	1	Linear	$2048 \rightarrow 128 \times 4 \times 4$ (shaped)		LeakyReLU
	2	ConvTranspose2D	$64 \times 8 \times 8$	kernel=4, stride=2, padding=1	
	3	BatchNorm2D		eps=1e-5, momentum=0.1	ReLU
	4	ConvTranspose2D	$64 \times 16 \times 16$	kernel=4, stride=2, padding=1	
	5	BatchNorm2D		eps=1e-5, momentum=0.1	ReLU
	6	ConvTranspose2D	$64 \times 32 \times 32$	kernel=4, stride=2, padding=1	
	7	BatchNorm2D		eps=1e-5, momentum=0.1	ReLU
	8	ConvTranspose2D	$3 \times 64 \times 64$	kernel=4, stride=2, padding=1	Sigmoid

portant to note that the training process does not consider the noisyfication process.

Regarding the noisyfication procedure, there are two natural options:

- Adding Gaussian noise with zero mean and a variance σ^2 that determines the degree of feature obfuscation.
- Passing the binary classifier output through a symmetric binary channel with a crossover probability of p. In other words, the binary outputs of the classifier are randomly converted to each other with a probability of p.

The parameters σ^2 and p control the privacy level.

The utility provider and adversary structures are similar to the encoder plus classifier structure, with the only difference being the output size of the last layer. The output size depends on the number of private features the adversary aims to infer or the number of features desired by the utility provider for inference.

To measure utility and privacy, we assume that the adversary's goal is to infer the gender (male/female) feature from the obfuscated dataset. The adversary is trained to make this inference and subsequently applied to the obfuscated dataset. The accuracy of the adversary in correctly determining the gender is considered a measure of privacy. On the other hand, the smiling feature is considered the desired feature for the utility provider. We train the utility provider on the obfuscated dataset and then test the trained network on the original dataset. The accuracy of the utility provider in recognizing the desired feature (smiling) is regarded as a measure of utility.

Our method enables the utility provider or trusted authority to train an obfuscator for the private feature of concern. This obfuscator can be shared with data providers, relieving them of the training burden. Data providers can adjust the trade-off between utility and privacy by fine-tuning the noise level according to their desired level of feature privacy.

Algorithm 1 Training phase of the framework

```
Input: Training dataset samples $\mathbf{X}$
    Parameters: learning rate $\eta$
    $n_e$, $n'_e$, $n_b$, $n_w$
    loss weights $w_{ae}$, $w_{\mathsf{dcor}}$, $w'_{ae}$, and $w'_{\mathsf{dcor}}$
Output: Obfuscator Model
    Initialization.
 1: for $n_e$ epochs do
 2:     for $i = 0$ to $n_b - 1$ do
 3:         Randomly picked mini-batch of size $B$
 4:         $\beta_{i+1} = \beta_i - \eta \nabla_\beta L_{ae}$
 5:         $\phi_{i+1} = \phi_i - \eta \nabla_\phi L_{ae}$
 6:         $\theta_{i+1} = \theta_i - \eta \nabla_\theta L_{ae} - \eta \nabla_\theta L_{\mathsf{C}}$
 7:         $\alpha_{i+1} = \alpha_i - \eta \nabla_\alpha L_{ae} - \eta \nabla_\alpha L_{\mathsf{C}}$
 8:     end for
 9: end for
10: for $j = 0$ to $n'_e - 1$ epochs do
11:     if $j = n_w$ then
12:         $w_{ae} \leftarrow w'_{ae}$
13:         $w_{\mathsf{dcor}} \leftarrow w'_{\mathsf{dcor}}$
14:     end if
15:     for $i = 0$ to $n_b - 1$ do
16:         Randomly picked mini-batch of size $B$
17:         $\beta_{i+1} = \beta_i - \eta w_{ae} \nabla_\beta L_{ae}$
18:         $\phi_{i+1} = \phi_i - \eta w_{ae} \nabla_\phi L_{ae} - \eta w_{\mathsf{dcor}} \nabla_\phi L_{\mathsf{dcor}}$
19:     end for
20: end for
21: return
```

4 Performance Evaluation

Our experiment is performed using PyTorch [34] on an NVIDIA 1080 Ti GPU. The image size is 64×64×3, and a mini-batch technique is employed with a batch size of $B = 64$. For training all networks, we utilize the Adam optimizer [35] with a learning rate of 0.001. The obfuscator, adversary, and utility provider training procedures are conducted with similar configurations. Networks' layers start with random weights from Gaussian distributions: mean zero, variance 0.02 for convolutions and fully connected layers, and mean one, variance 0.02 for batch normalization layers. To mitigate overfitting, a dropout technique is implemented in certain layers by randomly excluding units with a probability of 0.5. The obfuscator undergoes training on the CelebA dataset for several epochs. The plot in Figure 4 shows the Peak Signal to Noise Ratio (PSNR) between $\mathbf{X}$ and $\mathbf{X}'$ as a function of epochs. Afterward, the output of the classifier is perturbed by either flipping its output with a probability of p or adding Gaussian noise with variance σ^2. The entire dataset $\mathcal{D}$ is then transformed into the dataset $\mathcal{D}'$ through this obfuscation process.

As depicted in Figure 4, notable changes can be observed in the autoencoder training loss during specific epochs, particularly at epoch numbers 50 and 87. During these epochs, there is a sudden increase in the autoencoder loss, which can be attributed to the adjustment of the loss function weights at the beginning of these epochs. Regarding the earlier fluctuations in the autoencoder loss occurring prior to epoch number 50, they arise from inherent characteristics associated with employing the Adam optimizer.

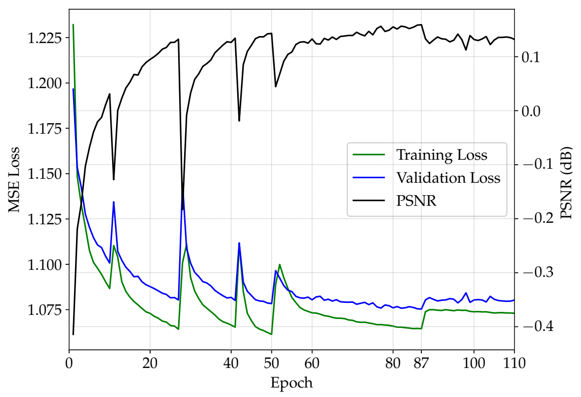

Figure 4. Variation of autoencoder loss and PSNR with respect to the number of epochs

When we consider the obfuscated dataset created by our scheme, employing flipping noise with a probability of 0.5, we observe a significant enhancement in privacy with only a minimal reduction in utility. To illustrate, let us assume an adversary that has been well-trained on the CelebA dataset to extract gender information. When this adversary attempts gender inference on the obfuscated test set, it achieves an accuracy of 0.756. In contrast, the accuracy of gender inference on the unobfuscated test set is notably higher at 0.988. This contrast highlights the substantial privacy amplification provided by the proposed algorithm, which amounts to an improvement of 0.232. Shifting our focus to utility, the accuracy of a user's inference regarding the smiling feature from the obfuscated test set stands at 0.85, in comparison to 0.95 for the original test set. This slight reduction in utility reaffirms the effectiveness of the proposed method.

In Figure 5, we compare the utility-privacy trade-off capability of our proposed scheme with various methods from different references, where the points on their curves are obtained from [21]. Considering gender as the private feature and smiling as the non-private feature, we depict the trade-off curves using both flipping and Gaussian noisifications, demonstrating superior performance compared to the other methods. For the flipping case, the points progress from right to left, representing the probabilities $p = 0.1, 0.2, 0.3, 0.4, 0.5$. On the other hand, in Gaussian noisification, the points are plotted for noise variances $\sigma^2 = 5\nu, 10\nu, 15\nu, 20\nu, 50\nu$. Here, ν denotes the mean

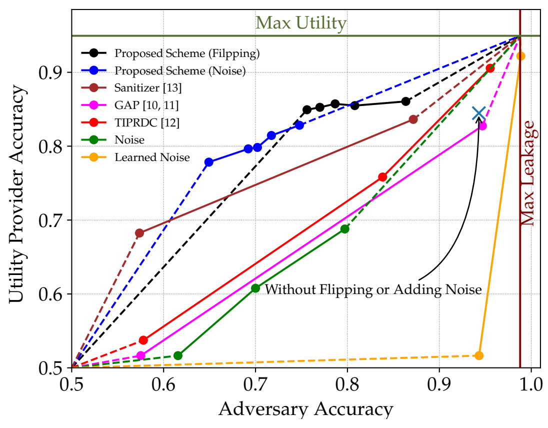

Figure 5. Comparing the trade-off between utility and privacy in the proposed scheme to that of existing schemes

value of the output nodes from C for each record. Additionally, it is evident from the figure that by employing both the methods of adding Gaussian noise and flipping, we can effectively support a wide range of utility and privacy requirements. This provides data owners with a greater degree of freedom in adjusting the trade-off between utility and privacy by adding Gaussian noise or flipping. Furthermore, Figure 5 illustrates the maximum achievable accuracy for both the adversary and the utility provider, which are respectively trained to extract gender and smiling features.

The references for the algorithms used in the comparison are provided in the figure. The term “Noise” refers to the addition of Gaussian noise with a mean of zero and a variance of 40, as implemented in [20]. In the learned noise method, a noise component is introduced to a DNN, and the resulting output is added to the dataset [18, 19]. To ensure a fair comparison, both the adversary and the utility provider are designed to closely align with the previous works in terms of achieving maximum privacy and utility.

The visual impact of obfuscation on a set of images can be observed in Figure 6. As can be seen, increasing flipping and Gaussian noises lead to the gender feature becoming more vague, while the smiling feature is preserved.

So far, we have regarded gender as a private feature and smiling as the desired feature for the utility provider. The extent of correlation between the private feature and the desired feature of the utility provider is anticipated to have an impact on the utility derived from the obfuscated dataset. To explore the influence of this correlation, we initially identified several features that exhibit varying degrees of correlation with the private feature of gender. We employed the same correlation measure as described in (9) to

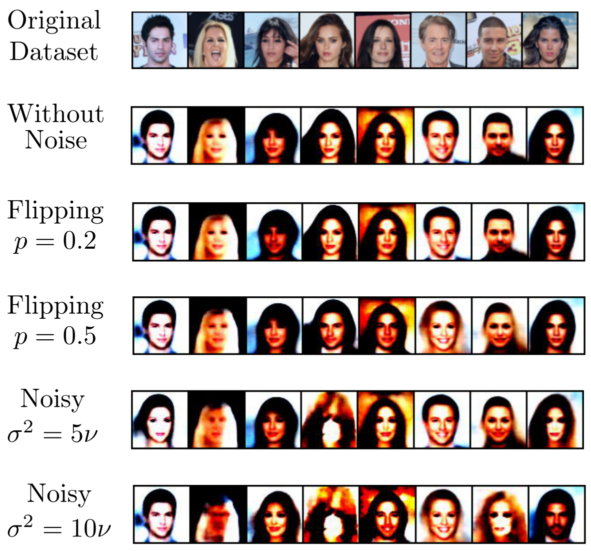

Figure 6. The images include originals, noise-free generated ones, and obfuscated versions with varying flipping probabilities and Gaussian noise variances. Here, ν represents the mean value of the output nodes from C for each record.

compute these correlations. Specifically, we computed the correlation between 40 features in the CelebA dataset and the private feature of gender. Among these features, we selected four (wearing lipstick, attractive, chubby, and pale skin) that displayed different levels of correlation, ranging from almost no correlation to a very strong correlation. These features exhibited correlations of 0.84, 0.4, 0.1, and 0.05 with gender, respectively. Notably, the correlation of two of these features is lower than the correlation between the smiling feature and gender, which is 0.17, while the other two have higher correlations. Substituting these four alternative desired features in place of smiling, we present the utility-privacy trade-off curves in Figure 7, taking into account flipping noise with probabilities of 0.1 and 0.5, as well as Gaussian noise with variances of 5ν and 50ν. To ensure a fair comparison among different features, we normalize the achieved utility values for each feature to the maximum utility achievable for that feature on the original dataset. As anticipated, the figure illustrates that when the correlation between the private and desired features is lower, the reduction in utility with increasing noise is less pronounced, resulting in a curve with a gentler slope. Moreover, within a significant range of correlations, the proposed obfuscation method performs well. However, in cases of extreme correlations, such as wearing lipstick, it is expected that the impact of adding noise on utility becomes more noticeable.

In Figure 8, the autoencoder and decorrelation losses are depicted versus epochs. As you can see, any decrease in one loss results in a controlled increase in another loss, highlighting the importance of the

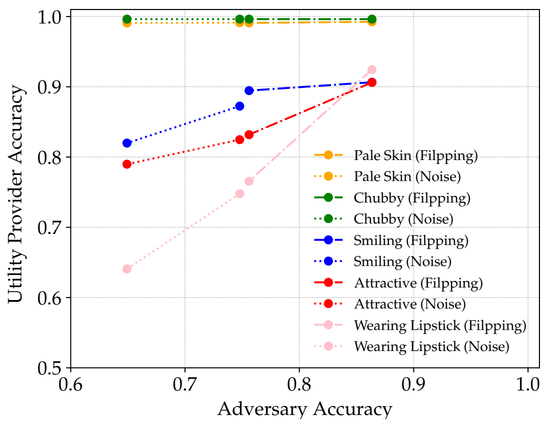

Figure 7. The impact of the correlation between desired and private features on the utility-privacy trade-off, taking into account flipping and Gaussian noises

carefully designed weighting process in Algorithm 1.

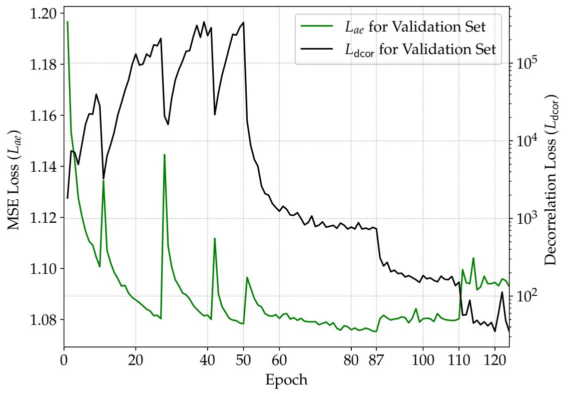

Figure 8. The trend of autoencoder and decorrelation losses with respect to the number of epochs

5 Exploring Additional Privacy Considerations

In this paper, we designed our adversary and utility provider to closely resemble those used in previous works, with the goal of achieving comparable levels of utility-privacy measures. Specifically, the structures of the adversary and utility provider consist of an encoder and classifier within our obfuscator scheme, as illustrated in Table 1. In the following, we address two additional privacy concerns related to centralized learning.

- **Backdoor existence:** This concern revolves around the trustworthiness of the entity responsible for training the obfuscator, often referred to as the “trusted center”. The worry is that this trusted center might introduce a backdoor or other vulnerabilities into the system, which could compromise privacy. While this is a valid concern, it is acknowledged that trust assumptions are common in centralized learning scenarios, and they can be addressed through regulatory measures and the implementation of open systems accessible to all.
- **Sequential publishing attacks:** This concern is related to the potential threat of adversaries piecing together information from separate, sequentially published datasets to reveal private data. While securing each individual dataset may be possible, the combination of these datasets can still lead to privacy breaches. Fortunately, our framework demonstrates resilience against such attacks, owing to the utilization of large and diverse datasets like CelebA during the construction of our obfuscator. This significantly reduces the risk of privacy compromise in sequential publishing scenarios.

6 Conclusion

This paper has introduced a novel autoencoder-based framework that offers data providers the ability to protect their desired features’ privacy while preserving the dataset’s overall utility for other features. Our proposed structure not only outperforms similar approaches in terms of the utility-privacy trade-off but also stands out for its simplicity in design. Furthermore, we have examined the impact of the correlation between desired and private features on the performance of our scheme in the utility-privacy trade-off. As part of our future work, we aim to consider scenarios in which users possess various private features, with certain features potentially holding greater importance from a privacy perspective. Given this heterogeneous nature of features in terms of privacy concerns, we intend to extend the existing framework to extract the complete set of private features, decorrelate them appropriately, and obfuscate them using varying levels of noise. Another intriguing avenue for further research involves analyzing the proposed algorithm with differential privacy tools to provide an analytical assessment of the privacy offered by the scheme.

Acknowledgment

This work was partially supported by Iran National Science Foundation (INSF) under contract No. 97011231. This paper was presented in part at ISCISC 2023.

References

[1] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and*

communications security, pages 1310–1321, 2015.

[2] Hangyu Zhu, Rui Wang, Yaochu Jin, Kaitai Liang, and Jianting Ning. Distributed additive encryption and quantization for privacy preserving federated deep learning. *Neurocomputing*, 463:309–327, 2021.

[3] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[4] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[5] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[6] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, pages 480–501. Springer, 2020.

[7] Mojtaba Shirinjani, Siavash Ahmadi, Taraneh Eghlidos, and Mohammad R Aref. Private federated learning: An adversarial sanitizing perspective. *ISeCure*, 15(3), 2023.

[8] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

[9] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[10] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2006.

[11] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[12] Fatih Emekçi, Ozgur D Sahin, Divyakant Agrawal, and Amr El Abbadi. Privacy preserving decision tree learning over multiple parties. *Data & Knowledge Engineering*, 63(2):348–361, 2007.

[13] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[14] Anh-Tu Tran, The-Dung Luong, Jessada Karnjana, and Van-Nam Huynh. An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing*, 422:245–262, 2021.

[15] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[17] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[18] Chong Huang, Peter Kairouz, Xiao Chen, Lalitha Sankar, and Ram Rajagopal. Context-aware generative adversarial privacy. *Entropy*, 19(12):656, 2017.

[19] Peter Kairouz, Jiachun Liao, Chong Huang, Maunil Vyas, Monica Welfert, and Lalitha Sankar. Generating fair universal representations using adversarial models. *IEEE Transactions on Information Forensics and Security*, 17:1970–1985, 2022.

[20] Ang Li, Yixiao Duan, Huanrui Yang, Yiran Chen, and Jianlei Yang. Tiprdc: task-independent privacy-respecting data crowdsourcing framework for deep learning with anonymized intermediate representations. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 824–832, 2020.

[21] Abhishek Singh, Ethan Garza, Ayush Chopra, Praneeth Vepakomma, Vivek Sharma, and Ramesh Raskar. Decouple-and-sample: Protecting sensitive information in task agnostic data release. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIII*, pages 499–517. Springer, 2022.

[22] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R Rabiee. Deep private-feature

extraction. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):54–66, 2018.

[23] Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, 7(5):4505–4518, 2020.

[24] Hung Nguyen, Di Zhuang, Pei-Yuan Wu, and Morris Chang. Autogan-based dimension reduction for privacy preservation. *Neurocomputing*, 384:94–103, 2020.

[25] Bishwas Mandal, George Amariucai, and Shuangqing Wei. Uncertainty-autoencoder-based privacy and utility preserving data type conscious transformation. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[26] Mohammad Ali Jamshidi, Hadi Veisi, Mohammad Mahdi Mojahedian, and Mohammad Reza Aref. Adjustable privacy using autoencoder-based learning structure. *arXiv preprint arXiv:2304.03538*, 2023.

[27] Florian Tramer and Dan Boneh. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660*, 2020.

[28] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[29] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

[30] Samuel A Barnett. Convergence problems with generative adversarial networks (gans). *arXiv preprint arXiv:1806.11382*, 2018.

[31] Nisarg Raval, Ashwin Machanavajjhala, and Landon P Cox. Protecting visual secrets using adversarial nets. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1329–1332. IEEE, 2017.

[32] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Mohammad A. Jamshidi completed his undergraduate studies in Communication Systems at the University of Tehran in 2014. Subsequently, he pursued a Master of Science degree in Cryptography at Sharif University of Technology, completing it in 2016. In 2023, Ali earned his Ph.D. in Communication Systems from Sharif University of Technology. His research interests revolve around the domains of Deep Learning and Privacy-Preserving Learning.

Mohammad M. Mojahedian obtained dual B.Sc. degrees, one in Communications and the other in Electronics, from Amirkabir University of Technology (Tehran-Polytechnic), Tehran, Iran, in 2009 and 2010. He then pursued an M.Sc. degree in Communication Systems from Sharif University of Technology, Tehran, Iran, which he completed in 2012. In 2018, he successfully earned his PhD from the Communication Systems group at Sharif University of Technology. Since 2012, Mohammad has been affiliated with the Information Systems and Security Lab (ISSL) at Sharif University of Technology. Notably, he was a finalist for the Jack Keil Wolf student paper award at the ISIT symposium in 2015. His research interests encompass Information Theory, Wireless Communications, Statistical Learning Theory, and Computer Science. Currently, he is actively engaged in research collaborations with both Sharif University and Chalmers University of Technology.

Mohammad Reza Aref received the B.Sc. degree in 1975 from the University of Tehran, Iran, and the M.Sc. and Ph.D. degrees in 1976 and 1980, respectively, from Stanford University, Stanford, CA, USA, all in electrical engineering. He returned to Iran in 1980 and was actively engaged in academic affairs. He was a Faculty member of Isfahan University of Technology from 1982 to 1995. He has been a Professor of electrical engineering at the Sharif University of Technology, Tehran, since 1995. His current research interests include areas of Communication Theory, Information Theory, and Cryptography.